The importance of how-questions in technical domains

Schwitter, R ; Rinaldi, Fabio ; Clematide, S

Abstract: How-questions are difficult to answer automatically, since the answers are usually not available in form of a single string in a technical document but need to be constructed from various information located at different places in the document or even from external knowledge sources. In this paper, we show how procedural questions can be answered in the context of ExtrAns. ExtrAns is an answer extraction system and operates over technical documentations that are rich in procedural knowledge. This procedural knowledge is often expressed in a procedural writing style that follows more or less strict guidelines. These guidelines together with typographical conventions can be exploited in a systematic way to first construct and then extract informative answers to procedural questions.

Workshop Question – Réponse, TALN-04, Fès, 22 avril 2004

The Importance of How-Questions in Technical Domains

Rolf Schwitter (1), Fabio Rinaldi (2), Simon Clematide (2)
(1) Centre for Language Technology - Macquarie University
Sydney, NSW 2109, Australia
schwitt@ics.mq.edu.au
(2) Institute of Computational Linguistics - University of Zurich
CH-8057 Zurich, Switzerland
{rinaldi|siclemat}@cl.unizh.ch

Mots-clefs – Keywords

Questions Procédurales, Extractions de Réponses, Formes Logiques Minimales
Procedural Questions, Answer Extractions, Minimal Logical Forms

Résumé - Abstract

Répondre de façon automatique à des questions procédurales demeure difficile, car les réponses dans un document technique ne sont habituellement pas disponibles sous forme de chaînes uniques, mais doivent être construites à partir de différentes informations se trouvant à divers endroits dans le texte, et parfois même à partir de sources de connaissance extérieures. Dans cet article nous montrons comment répondre à des questions procédurales dans le cadre d'ExtrAns. ExtrAns est un système d'extraction de réponses qui peut être utilisé pour des documents techniques riches en connaissance procédurale. Cette connaissance procédurale est souvent exprimée dans un style d'écriture qui suit – plus ou moins – des règles strictes. Ces règles, associées à des conventions typographiques, peuvent être utilisées de manière systématique pour d'abord construire, puis extraire des réponses informatives à des questions procédurales.

How-questions are difficult to answer automatically, since the answers are usually not available in form of a single string in a technical document but need to be constructed from various information located at different places in the document or even from external knowledge sources. In this paper, we show how procedural questions can be answered in the context of ExtrAns. ExtrAns is an answer extraction system and operates over technical documentations that are rich in procedural knowledge. This procedural knowledge is often expressed in a procedural writing style that follows – more or less – strict guidelines. These guidelines together with typographical conventions can be exploited in a systematic way to first construct and then extract informative answers to procedural questions.

1 Introduction

In general, technical documentation is rich in procedural knowledge. Procedural knowledge is knowledge of *how* to perform a specific task or *how* to solve a problem. This procedural knowledge or know-how is in most cases either tailored to the specific needs of an expert user or a casual user of a technical device. In both cases this knowledge is very often described in a procedural writing style that consists of a set of ordered instructions potentially accompanied by examples and illustrations. Here is a real-world textual example from the car maintenance domain that instructs the user in a stepwise manner how to change the battery of a car:

1. Open the car hood.
2. Remove corrosion from battery with baking soda and water.
   Disconnect the negative battery cable.
3. Disconnect the positive battery cable.
4. Remove any battery hold downs.
5. Pick up battery out of car with battery handle.
   (If not equipped, use a battery carrying tool.)
6. Install new battery.
7. Replace all the hold downs.
8. Put on the positive cable.
9. Put on the negative cable.
10. Make sure everything is tight.

If this raw text is used as a knowledge source of an answer extraction system such as ExtrAns, then a question like

How do I change the battery?

can not be directly answered by simply looking up a string (or the logical form of that string) in the text as this might be the case for factoid (TREC-like) questions (16) such as

Where is the battery?

since the answer to this question might occur in the text as a single string denoting a specific location.

There exists a wide spectrum of question/answer complexity, whereas answers start out as simple facts but move to templated answers and then progress further to move to multimodal answers (8). Answers to *how*-questions fall somewhere in the middle of this spectrum.

Recent contributions to the QA track of TREC used the Web as a means to obtain data redundancy and avoid the need for complex linguistic analysis (2). The rationale is, provided that we have enough data, there will always be some passage that explicitly shows the answer to the question using a simple pattern. The Web becomes a knowledge resource that can be accessed by search engines and used for question answering. While it is not difficult to find on the Web many relevant sources of information for open-domain factoid type of questions, in the case of restricted technical domains this is much less likely. However, our original domain of experimentation (the Unix man pages) is an exception, as in general the domain of information

technology is well represented on the Web. Such circumstances allow us to experiment with fallback strategies based on Web search. But in many other technical domains (e.g. aviation domain) the situation is very different and the amount of information available on the Web is much scarcer, making such an approach unsuitable.

The reminder of this paper is structured in the following way: In Section 2, we first describe our original answer extraction system (ExtrAns) and then we discuss the specific problems of procedural questions in two different domains. In Section 3, we focus on the Unix man pages and show how we can use external knowledge to support the question answering process. In Section 4, we look at the Aircraft Maintenance Manuals (AMMs) of the Airbus A320 that are written in a controlled natural language and discuss how standardized typographical conventions and linguistic surface patterns can be exploited to improve the question answering process.

2 ExtrAns

ExtrAns is a question answering (QA) system specifically targeted at restricted domains, in particular technical and scientific domains rich in terminology (13). In contrast to other modern QA systems that operate over large collections of documents and use relatively little linguistic information, ExtrAns answers questions over technical domains exploiting linguistic and typographical knowledge from the documents and terminological knowledge about a specific domain. Various applications of the ExtrAns system have been developed, from the original prototype aimed at the Unix man pages to a version targeting the AMMs of the Airbus A320 (10). An evaluation of the question answering capabilities of ExtrAns against a baseline IR system is presented in (12).

The ExtrAns system analyses all background documents in an off-line stage and stores the semantic representation in a knowledge base. In an on-line stage (see Figure 1), the semantic representation which results from the analysis of the user query is logically matched against the stored representations in the knowledge base, locating those sentences that best answer the query.

In the off-line stage, the background documents are first processed by a tokenizer and terminology-processing module that identifies word and sentence boundaries and marks previously-identified domain-specific multi-word terms. A subsequent linguistic analyzer uses Link Grammar (15) to produce the syntactic structure of the sentences. Different forms of attachment ambiguities (prepositional phases, gerunds, infinitives, and *wh*-relative clauses) are resolved by an extension of Brill and Resnik's approach (1). Sentence-internal pronouns are dealt with using the anaphora resolution algorithm of Lappin and Leass (7). From these partially disambiguated syntactic structures, ExtrAns derives one or more logical forms as semantic representation for the core meaning of each sentence.

Terminology is particularly important in technical domains. The tokenizer detects the terms (previously extracted from the documents and collected in the thesaurus) as they appear in the input stream, and packs them into single lexical tokens prior to syntactical analysis (3). Another task performed by the tokenizer upon detection of a term is to replace it by its synset identifier (as stored in the thesaurus). In this way any term contained in a user query is automatically mapped to all its variants. This approach amounts to an implicit 'terminological normalization' and semantic disambiguation for the domain, where the synset identifier can be taken as a

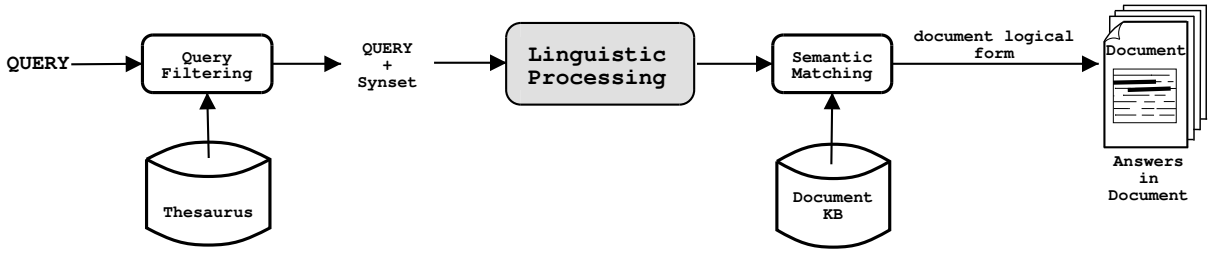

Figure 1: Schematic representation of the core ExtrAns system

reference to the 'concept' that each of the terms in the synset describe (5).

ExtrAns depends heavily on its use of minimal logical forms. ExtrAns' minimal logical forms are designed so that they are easy to build and to use, yet expressive enough for the task at hand. Not least importantly, the minimal logical forms and associated semantic interpretation method are designed to cope with problematic sentences. This includes very long sentences, even sentences with spelling mistakes, and structures that are not recognized by the syntactic analyzer. An additional advantage of ExtrAns' minimal logical forms is that they can be produced with minimal domain knowledge. This makes this technology easily portable to different domains.

Unlike sentences in documents, queries are processed on-line and the resulting minimal logical forms are proved by deduction over the minimal logical forms of document sentences stored in the knowledge base. When no direct answer for a user query can be found, the system is able to relax the proof criteria in a stepwise manner. First, hyponyms are added to the query terms. This makes the query more general but maintains its logical correctness. If no answers can be found or the user determines that they are not good answers, the system will attempt approximate matching, in which the sentence that has the highest overlap of predicates with the query is retrieved. The matching sentences are scored and the best matches are returned.

An example of the output of ExtrAns can be seen in Figure 2. When the user clicks on one of the answers provided, the corresponding document will be displayed with the relevant passages highlighted (14). Another click displays the answer in the context of the document and allows the user to verify the justification of the answer, as exemplified in Figure 3.

3 How-Questions and the Unix man pages

It comes with no surprise that a large proportion of questions that users pose to our online version of the ExtrAns system are *how*-questions having the following form:

How do I create a symbolic link?

How can I delete a directory?

These questions are challenging, since in most cases an informative answer can not be extracted as a contingent string from the Unix man pages but needs to be constructed from information that is located at various places in the document and sometimes also in other knowledge sources. An ideal answer to such a *how*-question that naturally leads to a problem-solving process (or a previous decision-making process in case of multiple answers) should have the following form:

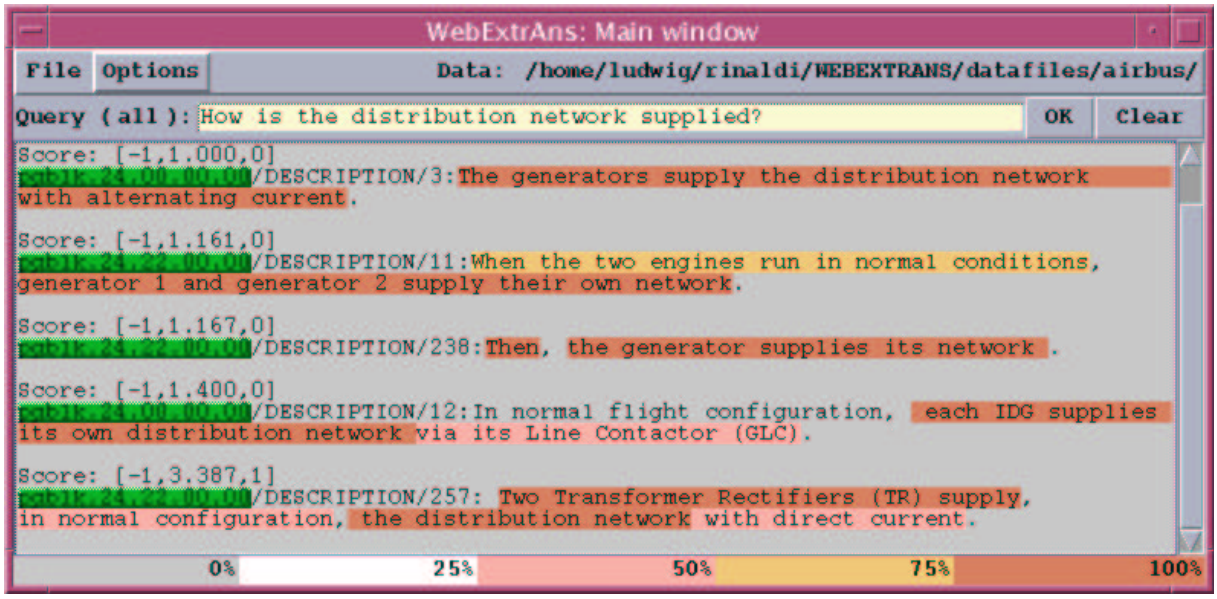

Figure 2: Example of interaction with ExtrAns

> *To create a symbolic link, use the* **ln** *command.*
>
> *For example, enter:* **ln -s <source> <target>**.

This answer provides instructions in a procedural writing style and illustrates the solution by an example. So far, it was not possible in ExtrAns to generate such informative answers – although this would be highly desirable from a user's perspective.

In the following discussion, we will show that a solution towards such informative answers is viable with a simple extension of the current framework. We combine minimal logical forms with pattern-matching techniques and Web-based knowledge mining whenever the Unix man pages do not provide enough information. Note that the use of surface pattern plays a complementary role in the overall question answering strategy.

First, let's start our discussion with a closer look at ExtrAns' current processing strategy for *how* questions and then describe the extension. In ExtrAns, the processing of a *how*-question such as

> *How do I create a symbolic link?*

results in a minimal logical form consisting of a number of predicates:

```
prop(symbolic,A,B),
object(s_link,C,[B]),
evt(s_make,D,[E,B]),
object(F,G,[E|H]).
```

In this flat notation, words like *link* and *create* that are defined in a thesaurus together with a list of synonyms have been replaced by the terms `s_link` and `s_make` which denote entire synonym sets. The inference engine of ExtrAns unifies these generalized predicates that contain variables for concepts (e.g. `C` or `D`) and individuals (e.g. `B` or `E`) with the minimal logical forms

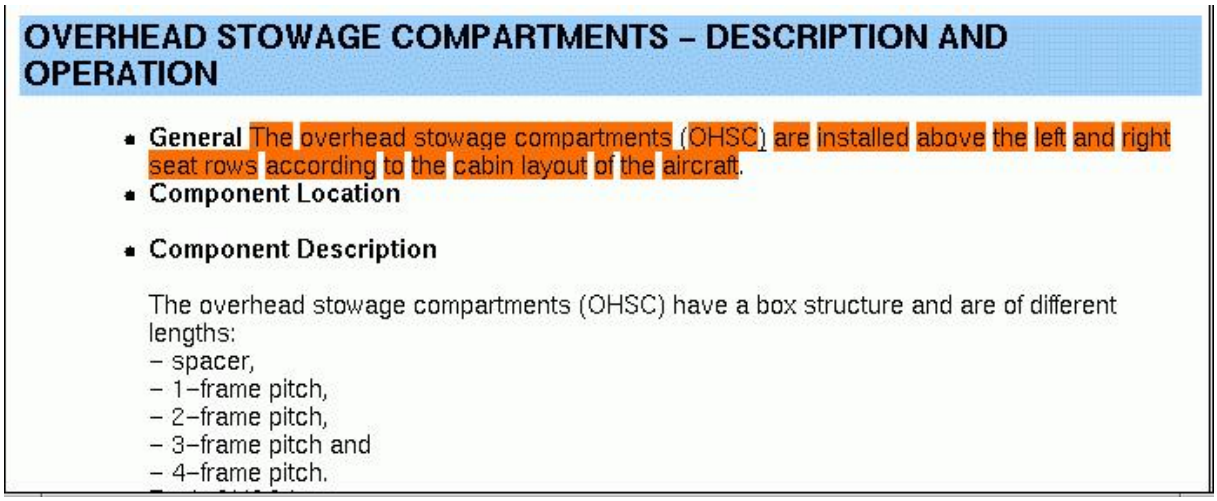

Figure 3: An example of ExtrAns' output

of document sentences that have been generated in the off-line stage. Note that all readings for ambiguous document sentences are asserted and evaluated – if no direct logical match is possible, then the fallback strategy swings into action that relaxes the proof criteria (see Section 2 for details).

For example, the following minimal logical form has been derived from a document sentence in an off-line stage:

```
holds(e4)~[3],
object(ln,a2,[x1])~[1],
object(s_command,a3,[x1])~[1],
evt(s_make,e4,[x1,x5])~[3],
object(s_link,a6,[x5])~[7],
prop(hard,p7,x5)~[4],
prop(symbolic,p8,x5)~[6],
to(x5,x10)~[8],
object(s_file,a9,[x10])~[9].
```

This logical form subsumes all predicates (displayed in bold face) that are available from the example question. This subsumption relation between document sentence and question is interpreted as a (partial) answer to the question. As the example shows, each predicate has an additional pointer (for example, `~[1]`) that refers to the surface form from which the predicate has been derived. These pointers make it possible to localize an answer in the Unix man pages and extract it:

ln - make hard or symbolic links to files

Unfortunately, this answer does only provide a partial solution to the initial *how*-question and does – without much doubt – not solve the problem of the user. The answer merely informs the user about the existence of a command but does not offer an explanation about *how* to solve the problem at hand; that means *how* to create a symbolic link.

But nothing is lost here, the important thing is that this partial solution provides information that can be exploited in a second step. Since command names are marked typographically in the Unix man pages, this information has been added to the initial minimal logical form for the document sentence. The following two predicates together express that **ln** is a Unix command:

```
object(ln,a2,[x1])~[1]
object(s_command,a3,[x1])~[1]
```

Most Unix man pages contain examples that explain how these commands are used in the *Notes* section of the man pages. Since these commands are based on a formal language of their own, they have a unique format with a well-defined argument structure and occur always with the same system prompt **example%**:

```
example% ln -s dir link
```

ExtrAns represents such information as a sequence of keywords, since the parser is not able to allocate a dependency structure to this string:

```
keyw('example%'), keyw('ln'), keyw('-s'),
keyw('dir'), keyw('link')
```

To generate informative answers, we first need to enrich the logical form for *how*-questions by adding information about the query type. Procedural questions such as *How do I ...* or *How can I ...* are assigned the query type **how_proc** resulting in the following logical form:

```
query(how_proc,D),
prop(symbolic,A,B),
object(s_link,C,[B]),
evt(s_make,D,[E,B]),
object(F,G,[E|H])
```

In a first step, ExtrAns' inference engine will extract for this query type the first line of the informative answer below which corresponds to a partial logical match. In a second step, ExtrAns will try to find one or more examples on the same man page. Since ExtrAns knows that **example%** is a system prompt and that **ln** is a command name and finds in the man page the information that the option **-s** creates a symbolic link, it is easy to construct an informative answer with the correct example that is close to our initial requirements:

> **ln** - **make** hard or **symbolic links** to files
>
> For example, enter: **`ln -s dir link`**

Unfortunately, not all Unix man pages come with a set of examples for each Unix command. In this case, we can use a fallback strategy: If ExtrAns would only find the *first line* of the above-mentioned answer in the Unix man pages but not the Unix command on the *second line*, then ExtrAns can try to acquire this missing information from an external document using the Web. Since ExtrAns already knows the command name **ln**, it can use this information in an external metasearch. For this purpose, the initial query is enriched with the command name (**ln**) and an appropriate domain constraint (**unix**) that tells the metasearch engine that the search is a phrase search (with two keywords) about the Unix domain:

"How do I create a symbolic link" ln unix

We made the observation that this kind of 'Web mining' retrieves tutorial-like documents. We can now use efficient pattern-matching techniques to extract suitable examples of Unix commands from the search engine's answer list using the command name and the specific pattern for Unix commands. The complete informative answer can now be constructed by combining the information extracted from the Unix man page (for the first line) and from the external knowledge source (for the second line). Additionally, ExtrAns should add a hypertext link to the second line that tells the user where the external information comes from; this makes the correctness of the information verifiable.

4 How-Questions and the Aircraft Maintenance Manuals

Procedural writing provides a way to order instructions sequentially and disposes of a small set of recurrent linguistic surface patterns that indicate the communicative function of the text. Typographical conventions such as headings, font size, font style, and outline formats such as lists marked with numbers or bullets underline this communicative function and improve the readability of the document.

Increasingly often, technical documentations are written in accordance with an in-house style guide or even follow an industrial standard. For example, aerospace manufacturers are required to write aircraft maintenance documentation in an industry-wide standard. The AMMs of the Airbus A320 that we use as knowledge source in ExtrAns are written in AECMA Simplified English (4).

AECMA Simplified English is a controlled natural language and consists of a restricted vocabulary with approved words and employs about 60 writing rules, for example:

- Use only the active voice in procedural writing, and as much as possible in declarative writing.
- Give only one instruction per sentence unless more than one instruction is to be done at the same time.
- Use tabulation to shorten sentences that include lists of things.

AECMA Simplified English distinguishes two writing styles: declarative and procedural writing. Each section in the AMM starts with a declarative description of a technical unit and then describes operational and functional procedures that are necessary to maintain this unit.

Below is an excerpt of a procedural description taken from a section about the removal/installation of the Lavatory Smoke Detector:

Procedure
A. Removal of the Lavatory Smoke Detector

1. Disconnect the electrical connector 1WQ-A (2) from the smoke detector (1).
2. Put blanking caps on the disconnected electrical connectors.
3. Remove the screws (5) and the washers (4) from the smoke detector (1).
4. Carefully remove the smoke detector (1) with the packing (3) from the rear-wall (7).

This procedural description starts with a heading *Procedure* that locates the procedural information in the manual unambiguously. The subsequent subtitle *Removal of the Lavatory Smoke Detector* describes in short form what the consecutive instructions are about. All these instructions follow a strict surface pattern: The sentences are numbered and written in imperative form and only one instruction per sentence is provided. The sentences are short and most noun phrases are introduced by a determiner. Brackets are used in a well-defined way and the numbers in the brackets refer to items on a part list that are further described elsewhere in the document.

To answer a question such as

How do I remove the smoke detector?

ExtrAns produces minimal logical forms and computes the query type **how_proc** that tells us that the answer needs to be a procedural piece of knowledge. Since the AMMs are well-structured and have explicit headings that introduce procedural paragraphs, ExtrAns can preselect those paragraphs as knowledge sources to answer *how*-questions and ignore the rest. The logical form of the question is then unified with the logical forms derived from the subtitle of the paragraphs. If this process is successful, then all numbered sentences that communicate the instructions can be extracted together with the subtitle. However, in our example the subtitle contains a nominalization *removal of* of the corresponding verb *remove* in the question. There exists a systematic relation between this noun phrase and the verb: if somebody removes *X*, then there exists a removal of *X* and vice versa. The logical form generator of ExtrAns translates the noun phrase in the subtitle into the following two disjunctive predicates with the help of a handcrafted lexical resource that contains information about the structure of nominalizations (see also (9)):

```
or([object(removal,a3,[a1,x1]),evt(s_remove,e3,[a1,x1])])
```

The inference engine of ExtrAns selects the applicable predicate during the question-answering process. This example also shows the importance of terminology in technical domains: *smoke detector* is less specific than *Lavatory Smoke Detector*. ExtraAns uses a hierarchy of subtypes (and morphosyntactic information) to identify linguistic variants between two terms (11).

5 Conclusion

In this paper we presented a complementary approach to improve the processing of *how*-questions in technical domains. Answers to such procedural questions are rarely detectable as a coherent string in the document collection (as it might be the case for factoid questions). We argued that the clear structuring of the text in technical documents – arising from standardized writing guidelines – makes it possible to exploit text coherence and discourse structure to answer procedural questions in an informative way. We are convinced that high performance in the question answering task is best achieved through fusion of logic-based and pattern-matching techniques using external knowledge sources – where appropriate.

Acknowledgments

The ExtrAns project (1996-2000) was funded by the Swiss National Science Foundation (contracts 1214-45448.95 and 1213-53704.98) and by the Gebert Rüf Foundation (contract RS-043/98). The work described in the paper has been conducted while the first author was on a sabbatical at the University of Zurich.

Références

1. Brill E., Resnik P. (1994), A rule-based approach to prepositional phrase attachment disambiguation. In *Proc. of the 15th International Conference of Computational Linguistics (COLING '94)*, vol. 2, 998-1004, Kyoto, Japan.

2. Brill E., Lin J., Banko M., Dumais S., Ng A. (2001), Data-intensive question answering. In E.M. Voorhees and D.K. Harman (eds), *The Tenth Text REtrieval Conference (TREC-10)*, NIST.

3. Dowdall J., Hess M., Kahusk N., Kaljurand K., Koit M., Rinaldi F., Vider, K. (2002), Technical terminology as a critical resource. In *International Conference on Language Resources and Evaluations (LREC-2002), Las Palmas*, 1897–1903, 29–31 May.

4. The European Association of Aerospace Industries (2004), AECMA Simplified English, AECMA Document: PSC-85-16598. *A Guide for the Preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language*, January.

5. Kageura K. (2002), *The Dynamics of Terminology, A descriptive theory of term formation and terminological growth*. Terminology and Lexicography, Research and Practice. John Benjamins Publishing.

6. Katz B., Lin J., Loreto D., Hildebrandt W., Bilotti M., Felshin S., Fernandes A., Marton G., Mora F. (2003), Integrating Web-based and Corpus-based Techniques for Question Answering. MIT Computer Science and Artificial Intelligence Laboratory Cambridge, MA 02139. In *Proc. of the Twelfth Text REtrieval Conference (TREC)* .

7. Lappin S., Leass H.J. (1994), An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4), 535–561.

8. Maybury M.T. (2002), Toward a Question Answering Roadmap. The MITRE Corporation. *Technical Paper*, Bedford, MA 01730, November.

9. Macleod C., Grishman R., Meyers A., Barrett L., Reeves R. (1998), NOMLEX: A Lexicon of Nominalizations. In *Proc. of EURALEX'98*, Liege, Belgium, August.

10. Mollá D., Rinaldi F., Schwitter R., Dowdall J., Hess M. (2003), Answer Extraction from Technical Texts. *IEEE Intelligent Systems*, 18(4), 12–17, July/August.

11. Rinaldi, F., Dowdall J., Hess M., Kaljurand K., Koit M., Vider K., Kahusk N. (2002), Terminology as Knowledge in Answer Extraction. In *Proceedings of the 6th International Conference on Terminology and Knowledge Engineering (TKE02)*, 107–113, Nancy, 28–30 August.

12. Rinaldi F., Dowdall J., Hess M., Mollá D., Schwitter R. (2002), Towards Answer Extraction: an application to Technical Domains. In *ECAI2002, European Conference on Artificial Intelligence, Lyon*, 460–464, 21–26 July.

13. Rinaldi F., Dowdall J., Hess M., Mollá D., Schwitter R. (2004), Question answering in terminology-rich technical domains. In Mark Maybury, editor, *Accepted for publication in "New Directions in Question Answering"*. AAAI Press.

14. Schwitter R., Mollá D., Hess M. (1999), ExtrAns — answer extraction from technical documents by minimal logical forms and selective highlighting. In *Proc. Third International Tbilisi Symposium on Language, Logic and Computation*, Batumi, Georgia, September.

15. Sleator D.D., Temperley D. (1993), Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, 277–292.

16. Voorhees E.M. (2003), Overview of the TREC 2003 Question Answering Track. In *Proc. of the Twelfth Text REtrieval Conference (TREC)* .